Performance of ARQ Go-Back-N protocol in Markov channels with unreliable feedback: Delay analysis*

Michele Zorzi† and Ramesh R. Rao‡
†Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy
‡Department of Electrical and Computer Engineering, University of California San Diego, USA

Abstract—In this paper, an ARQ Go-Back-N protocol with time-out mechanism is studied. Transmissions on the forward and the reverse channels are assumed to be subject to Markovian errors. A recently developed approach based on renewal theory is further extended and the steady state number of packets in the ARQ system is evaluated. This quantity is one of two components that contribute to the delay in the overall system. Simulation results, that confirm the analysis, are also presented. Based on the delay analysis, the trade-off involved in the choice of the timeout parameter is identified and discussed especially in the context of a mobile radio channel.

I. INTRODUCTION

Two types of error control techniques have been used extensively to enhance the reliability of data transmissions. In Forward Error Control (FEC), redundancy is introduced in order to correctly decode a corrupted packet, and in Automatic Repeat reQuest (ARQ), erroneous packets are detected and their transmission requested [1].

With the resurgence of interest in wireless communication networks, it is important to be able to evaluate the performance of error control schemes under assumptions that accurately model the wireless communication channel. In particular, the analyses which have been done in the past, based on the assumptions of i.i.d. packet transmissions and perfectly reliable feedback, will not be applicable to situations where the errors in consecutive slots are not independent, and where the feedback information may get corrupted due to errors in the return channel. Therefore, although related papers have appeared in the past, performance evaluation of ARQ schemes over fading channels has received renewed interest.

In particular, the combined effect of dependent transmissions and erroneous feedback has been considered only by Kim and Jn [2], for the basic ARQ protocols, and by Cho and Un [3], for some more elaborate protocols. In these papers, the throughput analysis is done, by using the theory of renewal processes in the flow graph technique [4]. Delay analyses are relatively easy when the delay associated with each retransmission is a constant. In such cases, the average throughput and delay are directly related, and throughput and delay analysis are essentially the same. In the context of protocols in which the time between two successive retransmissions of the same packet is difficult to express, the delay analysis does not follow easily from the throughput analysis. The Go-Back-N ARQ protocol with timer control is one such example. For such protocols, only the throughput performance is available in the literature.

In this paper, we consider the Go-Back-N ARQ protocol with timer control, whose throughput performance was recently studied by [3]. In [5], it is shown that the throughput analysis given in [3] provides an upper bound, and the exact performance is evaluated via a Markovian analysis. This paper addresses the delay analysis of the same protocol. The new results are obtained through a non-trivial generalization of the model used in [5], and the exact delay performance is evaluated. It must be noted that, in order to accurately study the protocol performance, both throughput and delay are needed. The analysis presented, although developed here for a specific protocol, is applicable to the performance study of other protocols with memory, for which standard techniques are not feasible and the flow graph approach is impractical.

In the following, the channel model (Section II) and the protocol (Section III) are described. The proposed technique, based on a Markov chain approach, is described in Section IV, and is applied to the study of the protocol in Section V. Finally, some numerical results are discussed in Section VI.

II. CHANNEL MODEL

In this paper, the channel is modeled as in [3], i.e., as a Gilbert channel in each direction. Thus the patterns of packet and feedback errors follow two independent first-order Markov models, which are described by the transition matrices

\[ M_F = \begin{pmatrix} p & q \\ r & s \end{pmatrix}, \quad M_B = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, \]

where \( p = 1 - q \) and \( r = 1 - s \) is the probability that the forward slot is successful given that the forward slot \( i - 1 \) was successful (unsuccessful), and similarly for the entries of \( M_B \), with reference to the backward channel. Note that \( 1/r \) and \( 1/c \) represent the average lengths of the bursts of errors, which are described by geometric r.v.'s.

We assume, in the following, that the packet length is a constant, equal to one time unit. The round-trip delay from the beginning of a transmission to the reception and decoding of the corresponding feedback information is \( m \) slots. Hence if a packet transmitted in slot \( i \) is negatively acknowledged, it will be retransmitted in slot \( i + m \). Positive (ACK) and negative (NAK) acknowledgements can never be confused with each other, i.e., the effect of backward errors is to map the ACK and NAK symbols to an Erasure symbol. Also, we assume that each ACK/NAK carries the information about all previous transmissions. This implies that a packet whose ACK is lost may be subsequently acknowledged by feedback received in the future.

*This work has been partially supported by CNR and MURST, Italy.

0-7803-2955-4/95/$4.00 © IEEE
III. PROTOCOL DESCRIPTION

In the following, we consider the Go-Back-N (GBN) ARQ protocol with timer control, as described in [3]. The receiver follows the standard rules of ARQ GBN [9], i.e., it sends an ACK for every correctly received packet. When it receives an incorrect packet, it sends a NAK instead, and discards every successive packet, until a correct copy of the negatively acknowledged packet is received.

The transmitter acts according to the following rules. It sends packets in order, as long as it receives ACKs on the backward channel. Upon reception of NAKs, it goes back to packet i, retransmitting in order all packets from packet i. If the feedback about packet i is detected in error, it is ignored. If it was an ACK, it is possible that a future ACK/NAK will provide information that may acknowledge packet i. In fact, the ACK/NAK of packet i contains implicit acknowledgement of all packets i < k. However, it is possible that the lost feedback was a NAK (in which case no more feedback will be sent, and the uncertainty could last for ever), or that the feedback channel is undergoing a very long burst of errors, so that a large number of ACKs get lost. To cope with such cases, a time-out mechanism is used. This is provided by the use of a counter, which causes the transmitter to retransmit packet i after t slots (and all packet following i afterwards), if by that time it is still unknown whether or not it was correctly received. This allows the transmitter to avoid deadlock or buffer overflow. Note that the transmitter buffer needs to retain only t packets. Of course, it must be t ≥ m. If t = m we have the classic GBN scheme, in which if an ACK is not received at the proper time, retransmission is immediately performed.

IV. DELAY ANALYSIS

A. Definitions

In order to track the protocol evolution, one must clearly distinguish between three types of packets.

- **Outstanding packets** are packets that have been transmitted but whose feedback is as yet unavailable. Note that the time-out expiration implies a decision about the transmission outcome: as a result, a packet can not be outstanding for more than t consecutive slots.

- **Packets in the system (PITS)** are packets whose transmission has occurred, but which are not necessarily outstanding. Note that an outstanding packet is in the system, but not all packets in the system are outstanding. Consider the following example: let the system be in the state in which NAK1 was lost. Since the receiver will not send any other feedback information, this situation will last until the timeout associated with packet 1 expires, t slots after it was transmitted. By that time, the transmitter will have transmitted t – 1 more packets (2 to t) after packet 1. Upon timeout expiration, the transmitter goes back to packet 1 and retransmits it. By the time ACK1 is expected (m – 1 slots later), there will be m outstanding packets (in fact, m is the minimum number of outstanding packets), i.e., packets 1 to m, whereas the number of packets in the system will still be t (i.e., packets 1 to t), until some correct receptions occur and this number decreases. Furthermore, if the number of packets in the system is greater than m, upon a successful reception (i.e., when a packet actually departs from the system) an “old” packet (one that is already in the system) will be transmitted.

- **Pending packets** are outstanding packets which have been transmitted at least m slots earlier. Note that pending packets are outstanding packets whose feedback is expected but has not been received, due to failures in the return channel. Note also that the difference between the number of outstanding packets and the number of pending packets is always m.

B. Extended Markov chain

The Markov chain approach to the delay analysis is more complex than the one for throughput evaluation in [5]. In the throughput analysis of [5], four states were used to track the outcome of the transmission on the forward channel and the corresponding feedback transmission on the reverse channel. These two transmissions are of course separated in time by the round trip delay. The four states were as follows: Z1, corresponding to an erroneous packet and a correct feedback transmission (NAK); Z2, corresponding to both transmissions being correct (ACK); Z3, corresponding to an erroneous packet and erroneous feedback; Z4, corresponding to a correct packet transmission and a lost ACK.

State transition probabilities and the sojourn time in each state were identified. This led to a semi-Markov representation, with transitions weighted by different delays, as in the treatment in [10]. The number of correctly received packets (rewards) was tracked by counting visits to the states corresponding to successful transmissions (i.e., Z2 and Z4). By Little’s theorem [9], the average delay is given by the average number in the system divided by the average throughput. Thus, one way to compute delay is to “integrate” over a certain time the number of PITS and divide the result by the number of packets which were successfully received and acknowledged during that time.

If we take these measurements over an infinite interval, and ergodicity holds, then we can obtain the ensemble averages of these quantities.

Thus, to compute delay, we must keep track of sojourn times, transitions and rewards as well as the number of packets which have already been transmitted once and await retransmission. It is intuitively clear that the number of such packets will be lower-bounded by m and upper-bounded by t. Consequently, a stay of m slots in state 1 could contribute from a minimum of m to a maximum of mt. Splitting each state into a number of new states, each of which is further labeled by the number of PITS, resolves this uncertainty.

As defined before, PITS are those whose first transmission has already occurred and for which an acknowledgement (explicit or implicit) has not been received. The number of packets in the system is sampled at the instant immediately before an outgoing transition from that state occurs. At that time, the information about the outcomes on the two channels (which determines the destination state) is not known. Therefore, the number of PITS is an attribute of the originating state, and doesn’t depend on the destination state.

We also define the cumulative number of PITS (CPITS) associated with a transition as the sum of the number of PITS in each slot over the number of slots the system will stay in the destination. In general, this is not just the number of PITS multiplied by the time delay involved, since the number of PITS may increase as time goes by. As an example, consider the situation in which there are m packets in the system and assume that a NAK is lost. The transmitter will keep transmitting and finding no more “old” packets in the system, it will transmit new ones. Therefore, the number of PITS in the first slot after the feedback was expected will be m + 1, then m + 2, and so on, until the number of packets reaches t, at which time the timeout of the oldest packet in the system will expire, and the...
transmitter will go back to that packet and transmit it again.

In summary, for the delay analysis, the state representation must not only keep track of sojourn times, transitions and rewards but also the number of PITS. The computation of the PITS associated with each transition requires some careful bookkeeping.

In order to track the protocol evolution, we observe the following methodology: (1) each state will be marked according to the number of PITS, \( U \), immediately before feedback is expected (in the slot before the transition); (2) a transition with an associated reward will decrease \( U \) accordingly (provided that \( U \geq m \), see below); (3) a transition with no reward will keep \( U \) constant, except when \( U \) is equal to the number of outstanding packets. In this latter case, in fact, all packets in the system are outstanding, and in the next slot the transmission of a new packet is triggered, i.e., a new packet enters the system and \( U \) is increased. Note that \( U \) can increase only by one at a time, since multiple entrance is not permitted. (4) \( U \geq m \) always: whenever a transition that could lead to \( U = m - 1 \) occurs, a new packet will enter the system, and \( U \) will not decrease below \( m \); (5) each transition is marked with the transition probabilities and with the CPITS contributed by the stay in the destination state.

Note that each state has a maximum and minimum admissible value for the number of PITS. For example, state 3, which is entered after a timeout expiration, can only have \( U = t \), and need not be split. The other states in general will have to be split, so that state \( i \) will give rise to a number of states, \( i(U) \), where \( U \) takes all admissible values in that state. This enhanced state \( i(U) \) is used in this study to track the delay, whereas \( i \), a state of the original chain as given in [5], was adequate to study throughput. The state \( i \) will be referred to as a superstate.

Each state will still have four outgoing transitions. Two states in the same superstate will have as destination of their outgoing branches, states in a common superstate. This agrees with the fact that, in order to study the transition structure of the process, the splitting is not necessary. The transition matrix of the state chain will be related to the transition matrix of the superstate chain, in the sense that to each entry of the latter corresponds a block in the former. Moreover, in each such block, the nonzero transition probability entries will all be equal to the corresponding entry in the matrix of the superstate chain. The same is true for the delay and reward involved in each transition, which do not depend on the number of PITS, but only on the origin and destination superstates. On the other hand, the CPITS associated with each transition does depend on the number of PITS. This is the reason why the extended chain is needed.

V. PERFORMANCE OF GO-BACK-N

As in [5], we can define an appropriate semi-Markov process, to keep track of the quantities of interest. If \( R_{ij} \) and \( C_{ij} \) are respectively the number of rewards and the CPITS associated to a transition from \( i \) to \( j \), and \( R_{i} \) and \( C_{i} \) are the averages over the destination, \( j \), a fundamental theorem of renewal reward processes allows us to write [11]:

\[
D_p = \lim_{t \to \infty} \frac{R(t)}{C(t)} = \frac{\sum_{i=1}^{N} \pi_i R_i}{\sum_{i=1}^{N} \pi_i C_i},
\]

where \( \pi \) is the asymptotic distribution of the Markov chain, and \( D_p \) is the steady-state (i.e., average, if ergodicity is assumed) packet delay.

At this point, the problem is solved, assuming that we are able to actually compute the matrices and to solve for the stationary distribution. As already mentioned, this is not very difficult, as long as the number of states is not too large. It is usually easy to get results for chains of 100 to 200 states.

As an example, consider the Go-Back-N protocol with timer control, where \( m = 5 \) and \( t = 7 \) slots. The embedded Markov chain transition diagram, the transition probabilities and the delay and reward matrices are given in [5].

The number of states needed to adequately represent the channel conditions and the number of outstanding packet was found to be \( 4 + 2(t - m) = 8 \). We will split each of these states according to the possible values of the number of PITS, \( U \), allowed in it.

State 1 allows all values of \( U \) between \( m \) and \( t \). On the other hand, state 2 does not allow \( U = t \): in fact, if 2 is reached from any state with \( U = t \), it must have \( U \leq t - 1 \), since a reward is associated to the transition, and after that no new packet enters the system, because there are always old packets to be retransmitted (unless all pending packets are acknowledged, in which case the destination will have \( U = m + 1 \) anyway). As already observed, state 3 allows only \( U = t \), since it always involves the timeout expiration. This is true for states \( S_4, S_5, S_6 \) as well.

As to states \( S_i \), note the following. State \( S_0 \) is reached when an ACK is not received. As a result, in the slot during which the system will stay in \( S_0 \), another packet will be transmitted. This means that the minimum number of packets in the system just before leaving \( S_0 \) is \( m + 1 \), whereas the maximum is \( t \). For the same reason, the number of packets in state \( S_1 \) (which can be entered only from \( S_0 \), where \( U \geq m + 1 \)) must be at least \( m + 2 \), i.e., it has to be \( t \). The complete state vector is therefore given as follows (the same order of the entries will be used in the matrix representations):

\[
\begin{align*}
1, (5), 1, (7), & 2, (5), 2, (6), 3, (7), S_6, S_7, S_8, S_9, S_{10}, S_{11}, S_{12}. 
\end{align*}
\]

A. Expanded transition diagram

The composite information, including transition probabilities, transition times, rewards and CPITS, can be adequately described by a semi-Markov model with 12 states. According to the rules defined in the previous section, a transition involving some reward will lead to a state whose \( U \) decreases accordingly, under the constraint that \( U \geq m = 5 \). Therefore, from any state \( i(t) \) all transitions due to the reception of an ACK and involving one reward will lead to state \( 2(t - 1) \). Similarly, those from \( i(t - 1) \) will lead to \( 2(t - 2) = 2(m) \), and those from \( i(m) \) to \( 2(m) \), since the number of packets cannot be smaller than \( m \). Note, however, that transitions with multiple rewards can occur, for example, from \( S_6(t) \) to \( 2(m) \) (two rewards) and from \( S_7(t) \) to \( 2(m) \) (three rewards). In the latter case, all pending packets are acknowledged, and a new packet enters the system, so that the label of the destination is \( m = t - 3 + 1 \). On the other hand, transitions entering state \( 1(U) \) will come from states with \( U \) packets (i.e., through transitions without any reward), except for the case in which this state is reached from \( S_0(U + 1) \) or \( S_1(U + 2) \) and some reward due to the implicit acknowledgment of outstanding packets is earned. All transitions from \( 3(t) \) will lead to nodes with maximum number of packets (i.e., \( 1(t), 2(t - 1), 3(t) \), and \( S_0(t) \)), as happens for transitions exiting \( S_6(t), S_7(t) \) and \( S_8(t) \).

As to CPITS, note that transitions not corresponding to the timeout expiration will lead to a state where the number of pack-
ets remains constant during the stay. The number of slots of stay is also a constant depending only on the state, and therefore for these transitions it is easy to find the $C_{ij}$'s. For example, a transition into node $1(U)$ will involve $U$ packets for $m$ successive slots, and therefore will have a corresponding $C_{ij} = mU$. On the other hand, transitions to states $2(U)$, $S_0(U)$ and $S_1(U)$ involve $U$ packets for one slot, and therefore the CPITS is just $U$.

When the system arrives at state $S_2(t)$, the timeout expires, and the next feedback is expected $m$ slots later, during which period the number of PDTS will remain equal to $t$, giving rise to a CPITS of $mt$ slots. The same happens upon entering state $S_3(t)$, whereas in state $S_4(t)$ there will be $t - 1$ slots to the next expected feedback, and therefore we have a CPITS of $t(t - 1)$.

Finally, consider transitions to state $3(t)$. Those coming from all states with $U = t$ involve $t$ packets in the system on the first slot of stay in $3(t)$ and in all the following, so that the CPITS will be $t^2$. On the other hand, a transition from a state for which $U = m$ will correspond to the following situation: in the first slot of stay in $3(t)$ we have $m + 1$ PDTS, in the following $m + 2 = t$, and from then it will remain constant for the following $t - m$ slots. Also, if the previous state had $U = t - 1$, in the first slot of stay in $3(t)$ the number of PDTS will not increase, since there is still an "old" packet to be transmitted before a new packet enters the system in the next slot. In both cases, the CPITS involved is given by $t - 1$ packets in the first slot, and $t$ in the following $t - 1$, i.e., $t^2 - 1$.

Based on the above considerations, the CPITS matrix for the process with $m = 5$ and $t = 7$ is the following (the order corresponds to the vector (3)):

$$C = \begin{pmatrix} 25 & 0 & 0 & 0 & 48 & 6 & 0 & 0 & 0 & 0 \\ 0 & 30 & 0 & 5 & 0 & 48 & 6 & 0 & 0 & 0 \\ 0 & 0 & 35 & 0 & 6 & 49 & 0 & 7 & 0 & 0 \\ 25 & 0 & 0 & 5 & 0 & 48 & 6 & 0 & 0 & 0 \\ 0 & 30 & 0 & 5 & 0 & 48 & 6 & 0 & 0 & 0 \\ 0 & 0 & 35 & 0 & 6 & 49 & 0 & 7 & 0 & 0 \\ 25 & 0 & 0 & 5 & 0 & 48 & 6 & 0 & 0 & 0 \\ 0 & 30 & 0 & 5 & 0 & 48 & 6 & 0 & 0 & 0 \\ 0 & 0 & 35 & 0 & 6 & 49 & 0 & 7 & 0 & 0 \\ 25 & 0 & 0 & 5 & 0 & 48 & 6 & 0 & 0 & 0 \\ 0 & 30 & 0 & 5 & 0 & 48 & 6 & 0 & 0 & 0 \\ 0 & 0 & 35 & 0 & 6 & 49 & 0 & 7 & 0 & 0 \\ 25 & 0 & 0 & 5 & 0 & 48 & 6 & 0 & 0 & 0 \\ 0 & 30 & 0 & 5 & 0 & 48 & 6 & 0 & 0 & 0 \\ 0 & 0 & 35 & 0 & 6 & 49 & 0 & 7 & 0 & 0 \end{pmatrix}$$

(4)

VI. NUMERICAL RESULTS
The delay of the protocol under consideration has been studied numerically. In this section we discuss the results.

In Fig. 1, the average packet delay, $D_p$, is plotted vs. the average block error rate, $e$, for two values of $r$. As expected, an increase in $e$ produces a larger delay; also, note that, for the same value of $e$, long bursts (i.e., small $r$) yield better performance, since the errors tend to be clustered, and therefore are less harmful. An analogous results was found in [5] for the throughput performance.

Fig. 2 reports $D_p$ vs. the time-out parameter, $t$. Of course, when $t$ is not too small, a larger $t$ tends to increase the delay, and the curves are monotonically increasing. However, for large $r$, corresponding to poorer performance, the length of a burst of errors tends to be small: as a result, time diversity can be exploited, and the beneficial effect of a limited increase of $t$ outweighs the potential increase in delay, so that the overall performance turns out to be better. Note that the same effect (although more evident than here) was found in the throughput analysis.

The analysis presented here and that in [5] allow us to compute both throughput and delay for a given set of values of $e$, $r$, $m$ and $t$. In particular, it is possible to choose one of them as a parameter, and to plot $D_p$ vs. $\eta$ as the parameter varies. The results obtained allow us to simultaneously study how a parameter affects the performance, in terms of throughput and delay. As an example, in Fig. 3 we plotted $D_p$ vs. $\eta$ for $m = 5$, $e = 0.1$, and several values of $r$. The parameter which generates the curves is the time-out period, $t$. Depending on the situation, the throughput-delay characteristic may be monotonically increasing, as happens for $r = 0.1$; for higher $r$, the first portion of the curve has a negative slope. This means that both throughput and delay can be enhanced at the same time, which is an interesting result.

A. Mobile radio application
We consider a mobile radio fading channel as a real-world example of application of the above theory. As we did for throughput in [5], in Fig. 4 the delay, $D_p$, is plotted vs. the fading margin of the system, $F$, for $f_2 T = 0.02144$, $m = 5$, and for $t = 7$, 10 and 15. $T$ is the slot time, and $f_2$ is the Doppler frequency of the channel [12]. It is seen that, unlike for throughput, the delay performance is generally worse for
larger $t$. However, it must be noted that, even for moderate values of the fading margin (7-10 dB), the delay increase is negligible, whereas the throughput can be improved by about 10%.

The throughput-delay characteristics for fixed $F$ and parameter $t$ are shown in Fig. 5. It can be seen that, for low fading margin, an increase in throughput is accompanied by an increased delay. On the other hand, for large values of the fade margin, increasing $t$ can simultaneously improve the two performance measures. Also, even when throughput and delay are monotonically related, the delay performance is not very sensitive, at least for $F$ not too small: for example, for $F = 10$ dB, a 10% throughput increase can be obtained while accepting a delay increase of about 0.5 slots.

The combined throughput-delay analysis, never presented so far in the literature for the case of dependent errors and unreliable feedback, is a powerful tool for the performance study and for the design of ARQ systems.

**VII. Conclusions**

In this paper, we study a Go-back-N ARQ scheme which exploits time-diversity to recover from feedback errors. A dependent structure for the error processes on the two channels

---

**REFERENCES**


